# **I22-0518 Umer Farooq**

**README**

**Description:**

This C++ program simulates a simple virtual machine capable of executing instructions represented by op codes. It defines a set of op codes and provides functionality to load instructions into memory, execute them, and manipulate data stored in registers and memory.

**Op Codes:**

* HALT: Halts the program execution.
* ADD: Adds two values.
* SUB: Subtracts two values.
* LOAD: Loads a value from memory into a register.
* STORE: Stores a value from a register into memory.
* AND: Performs a bitwise AND operation.
* OR: Performs a bitwise OR operation.
* XOR: Performs a bitwise XOR operation.

**Memory Layout:**

The memory is represented as a vector of integers. Each instruction is represented by a sequence of three integers: OpCode, Operand1, and Operand2. OpCode specifies the operation to be performed, while Operand1 and Operand2 are the operands for that operation.

**Sample Programs:**

**Sample 1:**

* + This program loads a value from memory address 9, subtracts a value from memory address 10, and stores the result in memory address 11.
  + Memory: [LOAD, 9, 0, SUB, 10, 0, STORE, 11, 0, 25, 10, 0, HALT]
  + Initial Memory State: All memory locations are initialized to 0.
  + Expected Output: The program halts after execution. The final memory state should show the result of the subtraction operation stored in memory address 11.

**Sample 2:**

* + This program loads a value from memory address 9, performs a bitwise XOR operation with a value from memory address 10, and stores the result in memory address 11.
  + Memory: [LOAD, 9, 0, XOR, 10, 0, STORE, 11, 0, 25, 10, 0, HALT]
  + Initial Memory State: All memory locations are initialized to 0.
  + Expected Output: The program halts after execution. The final memory state should show the result of the XOR operation stored in memory address 11.

**Compilation and Execution:**

To compile the program, use the following command:

**g++ -o instruction\_simulator instruction\_simulator.cpp**

To run the program, execute the compiled binary:

**./instruction\_simulator**

**Outputs:**

* The program will execute each instruction in the sample programs, printing the memory contents before and after each execution.
* After execution, the program will halt, and the final memory state will be displayed.

**Note:** Uncomment sample1() or sample2() call in the main() function to execute the desired sample program.